PWS01: Powershell Basics Lab 0

Important: RESPECT the naming convention, follow this step by step procedure

Create a test environment

Use Remote Desktop to connect to the following machines. IP are given by your instructor.
10.120.233."ip1" (Server)
10.120.233."ip2" (Workstation)
10.120.233."ip3" (Workstation)

IP Config

Use the following IP Static configuration on the 3 machines.
❑IP address: 10.120.233.ip1-ip3
❑Subnet mask: 255.255.255.0
❑Default Gateway: 10.120.233.70
❑DNS:
- ❑ DC:127.0.0.1
- ❑ Workstations: 10.120.233.ip1

Set roles – join the domain

❑Rename the Server to PowerServer01
❑Run the DC promo Command (Missing files are in c:\temp)
❑Domain Name: PowershellX.course (X = number given by your instructor)

❑Rename 2 workstations in PowerWks01 & PowerWks02 + Restart
❑Place them the in the domain + restart

Install .net Framework on PowerWks01 only → create a management workstation with all tools

❑. RSAT

Change firewall policy settings

❑Disable your firewall via Administrative template – Network – Network Connection- Windows Firewall.

Users

❑Create 2 domain users: PwsUser01 & PwsUser02
❑Create a domain admin: PWSDomainAdmin

Install Powershell V2 on PowerWks01 ONLY (Only on XP Workstation)

Log as PWSDomainAdmin and install the following components:

❑Remove Powershell 1.0 (on XP)
❑Windows Management Core Framework (Windows PowerShell 2.0, WinRM 2.0, and BITS 4.0) in WindowsXP-KB968930-x86-ENG.exe

Firewall

❑Update your policies on PowerWks01 and PowerWks02 (gpupdate /force)
❑Check: Your firewall is shut down.

This page is intentionaly blank

PWS01: Powershell Basics Lab 1

1. Get a list of all commands containing the verb get.

   Get-Command –Verb Get

2. Get a list of all commands containing the noun item.

   Get-Command –Noun Item

3. Get the list of examples for the get-service cmdlet.

   Get-Help Get-Service -Examples

4. Get a list of aliases in PowerShell.

   Get-Alias

5. Get a list of methods that can be invoked for a service.

   Get-Service | Get-Member –MemberType Method

6. Create an alias for launching notepad.

   Set-Alias –Name np –Value notepad

7. Get a list of all commands an their syntaxes containing the verb start

   Get-Command –Syntax –Verb Start

8. Get the help about redirection.

   Get-Help about_redirection

9. Syntax for IF Command

   Get-Help about_if

10 . Which parameter(s) is/are compulsory in the « Get-Command » command ?

   None

11. Get a list of the properties of the Date object.

   get-date | get-member -MemberType properties

Powershell Basics Lab 2

1.Get a list of all services and their Name, Staus, DisplayName and ServiceType.

Get-Service | Format-List –Property Name, Status, DisplayName, ServiceType

2.Get a table of all services and their Name, Status, displayname and servicetype.Make sure everything is displayed completely .

Get-Service | Format-Table –Wrap –Autosize –Property Name, Status, DisplayName, ServiceType

3.Organize your service table by status.

Get-Service | Format-Table –Wrap –Autosize –Property Name, Status, DisplayName, ServiceType -GroupBy Status

4.Append the result of exercise 3 to "*c:\temp\services.txt"*. Use 2 methods.

Get-Service | Format-Table –Wrap –Autosize –Property Name, Status, DisplayName, ServiceType -GroupBy Status | Out-File –FilePath c:\temp\services.txt -Append

Get-Service | Format-Table –Wrap –Autosize –Property Name, Status, DisplayName, ServiceType -GroupBy Status >> c:\temp\services.txt

5.Send errors of the following command to "*c:\temp\errors.txt*" :
Get-Command –Name test

Get-Command –Name test 2> c:\temp\errors.txt

Powershell Basics Lab 3

1. List all files and folder in the Windows directory.
   Get-ChildItem –Path C:\Windows –Recurse
   ls C:\Windows –Recurse
   dir C:\Windows –Recurse

2. List all files and folder in the Windows directory but make sure the datas are displayed one page at time.
   Get-ChildItem –Path C:\Windows –Recurse | Out-Host -Paging

3. List hidden files.
   Get-ChildItem –Path C:\Windows –Recurse -Force

4. List all .log files in the Windows directory
   *Get-ChildItem –Path C:\Windows –Recurse –Name *.log*

5. Create a folder on the C: drive.
   New-Item –Path C:\Folder1 –ItemType Directory

6. Create a file in this folder.
   New-Item –Path C:\Folder1\file1.txt –ItemType File

7. Create a registry key in HKLM:\Software\Microsoft\Windows\CurrentVersion
   New-Item –Path HKLM:\Software\Microsoft\Windows\CurrentVersion\newKey

8. Rename your file.
   Rename-Item –Path C:\Folder1\file1.txt newName.txt

9. Copy your folder including all of its content.
   Copy-Item –Path C:\Folder1 –Destination C:\Temp\Folder1 –Recurse -Force

10. Delete the registry key you just created.
    Remove-Item –Path HKLM:\Software\Microsoft\Windows\CurrentVersion\newKey

11. Read the content of the Windows Updates log file and select the « added updates »
    Get-Content $env:windir\windowsupdate.log | Select-String "Added update"

Powershell Basics Lab 4

1..NET Objects

a. Create a new instance of the System.Diagnostics.EventLog class for the system log on your machine
$sysLog = New-Object –TypeName System.Diagnostics.EventLog –ArgumentList "System"

b. Display the log content by page
$sysLog.Entries | Out-Host -Paging

c. Clean the log
$sysLog.Clear()

2.COM Objects

a. Create a desktop shortcut using the COM object

i. Create the shortcut.
$WshShell = New-Object –ComObject Wscript.Shell
$link = $WshShell.CreateShortcut("$Home/Desktop/example.lnk")

ii. Add a link to notepad
$link.TargetPath = notepad
$link.Save()

b. Create a network drive to "\\server04\data" using the COM object
$WshNet = New-Object –ComObject Wscript.Network
$WshNet.MapNetworkDrive("z:","\\server04\data")

c. List the content of the c:\temp directory using COM object
$fso = New-Object -ComObject Scripting.FileSystemObject
$temp = $fso.GetFolder("c:\temp")
$temp.Files

3.Static classes

a. What is 24 to the power 5 ?
[System.Math]::Pow(24,5)

4.WMI Objects

a. List all drivers that are running (Win32_SystemDriver)
Get-WmiObject Win32_SystemDriver

Powershell Basics Lab 5

1.Get a table of all services and their name, Status, DisplayName and ServiceType. Organize your service table by status then by name.
Get-Service | Sort-Object –Property Status, Name | Format-Table –Property Name, Status, DisplayName, ServiceType –Autosize -Wrap

2.List all drivers that are running (Win32_SystemDriver)
Get-WMIObject Win32_SystemDriver | Where-Object {$_.State –eq "Running"} | Format-Table -Autosize

3.List all drivers that are running and have a startup type set to manual.
Get-WMIObject Win32_SystemDriver | Where-Object {$_.State –eq "Running" –and $_.StartMode –eq "Manual"} | Format-Table -Autosize

4.List all file in C:\temp with a size greater than 2MB
*Get-ChildItem -Path c:\temp | Where-Object {$_.Length -gt 1024*1024*2}*

5.List all file in C:\temp with a size greater than 2MB. Display only the name and the size in MB.
Get-ChildItem -Path c:\temp | Where-Object {$_.Length -gt 1024*1024*2} | Select-Object -Property Name, Length | ForEach-Object -Process {$_.Length = ($_.Length)/1024/1024); Write-Host $_.Name $_.length}

OR

Get-ChildItem -Path c:\temp | Where-Object {$_.Length -gt 1024*1024*2} | Select-Object -Property Name, Length | ForEach-Object -Process {$_.Length = ($_.Length)/1024/1024); $_}

Powershell Operators and Expressions Lab 1

1. Arithmetic operators

a. 2 + "123”
"2" + 123
2 + "abc"
"2" + "abc"

b. "abc" * 1
"abc" * 2

c. $a = 1,2,3
$a = $a * 2 "$a"

d. "123" / 4
123 / "4"
"123" /"4"

2. Assignment operators

a. Swap to variables in 1 line of code($a = 1, $b = 2)

$a , $b = $b , $a

b. Assign multiple variables in one operation

$a = 1
$b = $a + 1
$c = $a + 2

$c = ($b = (($a = 1) + 1)) + 1

3. Comparison operators

a. 01 -eq 001
01 -eq "001"
"01" -eq 001

b. "abc" -eq "ABC "
"abc" -ieq "ABC "
" abc" -ceq "ABC"

4. Compare collections

a. 1,2,3,1,2,3 -eq 2

b. 1,"2",3,2,"1" -eq "2"

c. 1,"02",3,02,"1" -eq "2"

d. 1,"02",3,02,"1" -contains "02"

Powershell Operators and Expressions Lab 1 (Suite)

5. Pattern matching

Use the –match operator and a regular expression

a. $town = ”brussels”, ”london”, ”paris”, ”copenhage”, ”helsinki”

1. To find a town beginning by letter « h » or « p »

$town –match ”^(h|p)”

→ paris,helsinki

2. To find a town containing “ha”

$town –match “ha”

→ copenhage

3. To find a town finishing by « s »

$town –match “s$”

→ brussels, paris

b. $number = ”12345”,”73777”,”61342”,”89899”,”67612”,”8b0ef” ”,”42”,

1. To find a sequence beginning with a number from 0 to 7

$number –match ”^[0-7]”

→ 12345, 73777,61342,67612

2. To find a sequence beginning with a digit

$number –match ”^\d”

→ all sequence

3. To find a sequence where second caracter is a not a digit

$number -match "^.\D ” or **” ^.[^0-9]”**

→ 8b0ef

4. To find sequences of 5 digits

$number -match '^\d{5,}‘

→ all sequence exempt « 42 »

c. $exception = ”dollar$”, ”dollars”

To find dollar written with the $ caracter

$exception –match ”dollar\$” → dollar$

d. Create a statement to determine if the given string is formatted like a ip address

$ip = ″ 10.120.253.22 ″

$ip -match " (25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\b "

See regex rules for details

Lab 8 : Managing the windows Environment

1. Start 3 powershell sessions.
Stop all powershell sessions except for the current session (use the variable $PID).

```
Get-Process | Where-Object { ($_.processname -eq "powershell") -and ($_.id -ne $PID)} | stop-process
```

2. Restart the spooler service.
```
Restart-Service spooler -force
```

3. List BIOS information.
Get a list of hotfixids for all installed hotfixes.
Display the user logged on to a specific computer.
```
Get-WmiObject -class Win32_bios
Get-WmiObject -class Win32_QuickFixEngineering | Format-Table -property hotFixId
Get-WmiObject -class Win32_ComputerSystem | Format-Table -property username
```

5. List the registry entries under the key
HKEY_LM\software\microsoft\windows\currentversion.
List entry in this key.
Create a new registry entry.

```
Import-Module PsRemoteRegistry
Get-Command –module PsRemoteRegistry

Get-ChildItem hklm:software\microsoft\Windows\Current\\Version
New-RegKey –key « software\microsoft\Windows\Current\\Version » -name Subkey -passthru
```

6. Create a folder. Create a text file in this folder containing a list of foldernames. Create a local user and give that user read permission on the folder.
Create a small PS script that will create a new folder for each folder listed in the textfile. The permissions for these folders should be the same as on the first folder that you created.

```
$folders = Get-Content « c:\folders.txt »
$acl = Get-Acl c:\folder1
ForEach($f in $folders)
                    {
                    New-Item « c:\$f » -itemType directory
                    Set-Acl « c:\$f »  -aclObject $acl
                    }
```

Lab 9 : Active Directory

a. Download and install RSAT Remote Server Administration Tools = KB 958830 = Admipak on a Windows 2008 platform)
b. Turn on the Active Directory Module for Powershell on Windows 7
    a. Control Panel
    b. Program and Features - Remote Admin Service Tools, AD DS,…..Active Directory Module for Windows Powershell
c. In Powershell Get a list of all modules installed on your workstation
d. Import the Active Directory module in Powershell.
    a. Import-Module ActiveDirectory
e. Get a list of all commands available for this module
    a. Get-Command -module ActiveDirectory

1. Get a list of all users in an OU (ex ou users in OU project100 in project100.trg)

```
Get-AdUser -filter * -searchbase "ou=users,ou=project100,dc=project100,dc=trg"
```

2. For each user, display only the name

```
Get-AdUser -filter * -searchbase "ou=users,ou=project100,dc=project100,dc=trg" | forEach-Object -process
    { $_.name }
```

3. Create a user with the following parameter:

- An encrypted password,name, description,displayname,GivenName,SamAccountName,UserPrincipal.

A domain admin password must be requested during execution of the script.

a: $password = convertTo-SecureString "password" -asPlainText -force

b: New-AdUser -Name "Automated User" -credential "project\administrator" -accountPassword $password -description "Use created with Powershell" -diplayname "User" -GivenName "Auto1" -UserPrincipalName "auto.bus1"

4. Enable the account you have just created

```
Enable-AdAccount -identity auto.b1
```

5. Create a group in Active Directory in the OU "project100" in the ou "group"

```
New-Adgroup -Name "PowershellGroup2" -SamACcountName Powershellgroup2 -groupcategory Security
    -GroupScope Global -DisplayName "PowerShell Group2" -path
    "ou=groups,ou=project100,dc=project100,dc=trg"
```

6. Create a script that create Users in Active Directory for a csv file.
If neccessary your instructor will provide you a csv file “users.csv”

```
# Import the Active Directory Powershell Module
Import-Module ActiveDirectory -ErrorAction SilentlyContinue

# Specify the target OU for new users
$targetOU = "OU=Users,OU=MyOrganization,DC=mil2008,DC=course"

# Find the current domain info
$domdns = (Get-ADDomain).dnsroot # for UPN generation

# Specify the folder and CSV file to use
$importedFile = "C:\scripts\CSV\Users.csv"

# Set the password for all new users
$password = read-host "Enter password" -assecurestring

# Parse the import file and action each line
$users = Import-CSV $importedFile

$users # to display on console

foreach ($user in $users)
{
$samname = $user.samaccountname
$dplname = $user.displayname
$givname = $user.givenname
$surname = $user.sn
$upname = "$samname" + "@" +"$domdns"

New-ADUser –Name $dplname –SamAccountName $samname –DisplayName $dplname -givenname
    $givname -surname $surname -userprincipalname $upname -Path $targetou –Enabled $true –
    ChangePasswordAtLogon $true -AccountPassword $password
}

#All done
```

LAB 9: Scripting

```
# Write a script
#   to ping all the IP adress of your domain
#   if the computer responds,
#   copy a presentation.txt on the local administrator's desktop
#   Create a log file with the "unsuccessful computer"


# select folder to copy
$app = new-object -com Shell.Application
$folder = $app.BrowseForFolder(0, "Select Folder", 0, "C:\")
$folderSource = $folder.Self.Path
if ($folderSource -ne "") {write-host "You selected "$folderSource}


# Define Workstation
# ------------------
$firstWks = read-host -prompt "Enter your FIRST workstation number - ex: 01"
$lastWks = read-host -prompt "Enter your LAST workstation number - ex: 17"

$firstWks..$lastWks | foreach {


      # Check if host name  $_ is < or > 10
      # -----------------------------------
      if ($_ -lt 10) {
                                          $number = "0$_"
                        } else {
                                          $number = "$_"
                        }

        # set workstation name
        $workstation = "powerwks$number"


      # check if workstation ping or not
      # --------------------------------

      $Ping = Get-WmiObject Win32_PingStatus -f "Address='$workstation'"
      echo " "

      if($Ping.StatusCode -ne 0) {

            echo ("$workstation is not pingable")

      } else {
            echo ("$workstation responds...")

            # copy bat file on host
            # ---------------------
            Copy-Item -Path $foldersource -destination "\\$workstation\c$\Documents and Settings\administrator\Desktop" -
recurse -force
            echo ("files copied...")

             }
    }
```